WRDC-TR-90-8007
Volume VII
Part 5

AD-A248 914

INTEGRATED INFORMATION SUPPORT SYSTEM (IISS)
Volume VII - Communications Subsystem
Part 5 - File Input/Output Primitives (FIOPS) Product
Specification

S. Barker

Control Data Corporation
Integration Technology Services
2970 Presidential Drive
Fairborn, OH 45324-6209

DTIC
SELECTE
APR 21 1992
S B D

September 1990

Final Report for Period 1 April 1987 - 31 December 1990

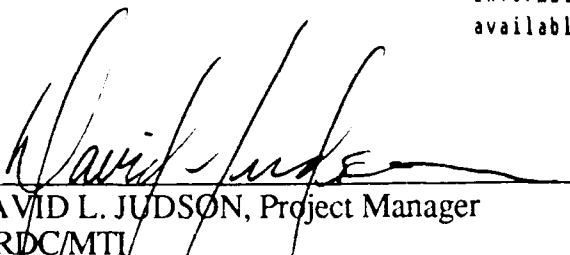Approved for Public Release; Distribution is Unlimited

92-09982

92 4 20 007

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever, regardless whether or not the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data. It should not, therefore, be construed or implied by any person, persons, or organization that the Government is licensing or conveying any rights or permission to manufacture, use, or market any patented invention that may in any way be related thereto.
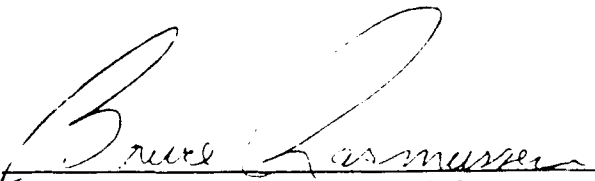
This technical report has been reviewed and is approved for publication.

This report is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations

DAVID L. JUDSON, Project Manager
WRDC/MTI
Wright-Patterson AFB, OH 45433-6533

DATE 25 July 91

FOR THE COMMANDER:

BRUCE A. RASMUSSEN, Chief
WRDC/MTI
Wright-Patterson AFB, OH 45433-6533

DATE 25 July 91

If your address has changed, if you wish to be removed form our mailing list, or if the addressee is no longer employed by your organization please notify WRDC/MTI, Wright-Patterson Air Force Base, OH 45433-6533 to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified		1b. RESTRICTIVE MARKINGS None		
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release; Distribution is Unlimited.		
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE				
4. PERFORMING ORGANIZATION REPORT NUMBER(S) PS 620343400		5. MONITORING ORGANIZATION REPORT NUMBER(S) WRDC-TR-90-8007 Vol. VII, Part 5		
6a. NAME OF PERFORMING ORGANIZATION Control Data Corporation; Integration Technology Services	6b. OFFICE SYMBOL (if applicable)	7a. NAME OF MONITORING ORGANIZATION WRDC/MTI		
6c. ADDRESS (City,State, and ZIP Code) 2970 Presidential Drive Fairborn, OH 45324-6209		7b. ADDRESS (City, State, and ZIP Code) WPAFB, OH 45433-6533		
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Wright Research and Development Center, Air Force Systems Command, USAF	8b. OFFICE SYMBOL (if applicable) WRDC/MTI	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUM F33600-87-C-0464		

8c. ADDRESS (City, State, and ZIP Code) Wright-Patterson AFB, Ohio 45433-6533	10. SOURCE OF FUNDING NOS.			
	PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
11. TITLE See block 19	78011F	595600	F95600	20950607

12. PERSONAL AUTHOR(S)
Structural Dynamics Research Corporation: Barker, S., et al.

13a. TYPE OF REPORT Final Report	13b. TIME COVERED 4/1/87-12/31/90	14. DATE OF REPORT (Yr.,Mo.,Day) 1990 September 30	15. PAGE COUNT 22

16. SUPPLEMENTARY NOTATION

WRDC/MTI Project Priority 6203

17.	COSATI CODES		18. SUBJECT TERMS (Continue on reverse if necessary and identify block no.)
FIELD	GROUP	SUB GR.	
1308	0905		

19. ABSTRACT (Continue on reverse if necessary and identify block number)

This specification establishes the detailed requirements for performance, design, test, and qualification of several computer programs collectively identified as File Input/Output Primatives (FIOPS).

BLOCK 11:

INTEGRATED INFORMATION SUPPORT SYSTEM
Vol VII - Communications Subsystem

Part 5 - File Input/Output Primitives (FIOPS) Product Specification

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED x SAME AS RPT. DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL David L. Judson	22b. TELEPHONE NO. (Include Area Code) (513) 255-7371	22c. OFFICE SYMBOL WRDC/MTI

DD FORM 1473, 83 APR   EDITION OF 1 JAN 73 IS OBSOLETE

FOREWORD

This technical report covers work performed under Air Force
Contract F33600-87-C-0464, DAPro Project. This contract is
sponsored by the Manufacturing Technology Directorate, Air Force
Systems Command, Wright-Patterson Air Force Base, Ohio. It was
administered under the technical direction of Mr. Bruce A.
Rasmussen, Branch Chief, Integration Technology Division,
Manufacturing Technology Directorate, through Mr. David L. Judson,
Project Manager. The Prime Contractor was Integration Technology
Services, Software Programs Division, of the Control Data
Corporation, Dayton, Ohio, under the direction of Mr. W. A.
Osborne. The DAPro Project Manager for Control Data Corporation
was Mr. Jimmy P. Maxwell.

The DAPro project was created to continue the development, test,
and demonstration of the Integrated Information Support System
(IISS). The IISS technology work comprises enhancements to IISS
software and the establishment and operation of IISS test bed
hardware and communications for developers and users.

The following list names the Control Data Corporation
subcontractors and their contributing activities:

SUBCONTRACTOR	ROLE
Control Data Corporation	Responsible for the overall Common Data Model design development and implementation, IISS integration and test, and technology transfer of IISS.
D. Appleton Company	Responsible for providing software information services for the Common Data Model and IDEF1X integration methodology.
ONTEK	Responsible for defining and testing a representative integrated system base in Artificial Intelligence techniques to establish fitness for use.
Simpact Corporation	Responsible for Communication development.
Structural Dynamics Research Corporation	Responsible for User Interfaces, Virtual Terminal Interface, and Network Transaction Manager design, development, implementation, and support.
Arizona State University	Responsible for test bed operations and support.

Table of Contents

SECTION 1

SCOPE

1.1 Identification

This specification establishes the detailed requirements for performance, design, test, and qualification of several computer programs collectively identified as File Input/Output Primitives (FIOPs). The FIOPs are authorized under Project Priority 6202 - Integrated Data Base Management (IDBM), Task 7003 - Common Data Model Runtime (CDMR). [2]

1.2 Functional Summary

This CPCI is used to allow the Common Data Model Runtime (CDMR) and possibly other Integrated Information Support System (IISS) programs a generic transportable interface to access system specific files.

This will allow common programs to be written on multiple host computers to perform their defined functions in the IISS.

A minimum set of functions are to be implemented in this task for sequential fixed-record-length files. These functions are identified as a Temporary File Namer, Open file, Read record, Write record, Seek one record forward or backward, Close file, and Sort/Merge file(s).

The FIOPs are needed for IISS programs in general because COBOL code for file access is not transportable in all cases. Also, the current CDMR programs on the VAX are written mainly in COBOL and they call file utilities written in FORTRAN. The subroutine calling conventions between COBOL and FORTRAN are the same on the VAX, but are different on the IBM. Therefore, it was determined that the FIOPs would be the best way to eliminate both of these problems.

SECTION 2

REFERENCES

2.1 Reference Documents

[1] ICAM Documentation Standards, 15 September 1983,
     IDS150120000A.

[2] Project Master Plan / Schedule: ICAM IDBM Project 6202,
     6 March 1986, PMP620220000.

[3] IISS System Requirements Document, 1 November 1985,
     SRD620140000.

[4] IISS System Design Specification, 1 November 1985,
     SDS620140000A.

[5] IISS CDM Subsystem Development Specification: Distributed
     Request Supervisor, 1 November 1985, DS 620141310.

[6] IISS CDM Subsystem Development Specification: Aggregator,
     1 November 1985, DS 620141320.

[7] IISS CDM Subsystem Development Specification: File Utilities,
     1 November 1985, DS 620141330.

[8] IISS Configuration Management: SCM Administrator's Manual,
     1 November 1985, CMA620124000.

2.2 Terms and Abbreviations

Aggregator - The IISS system (CDMR subsystem) process that merges
the files returned from a query into a consolidated result.

Application Program (AP) - A Cohesive unit of software that can be
initiated as a whole to perform some function or functions.

Common Data Model (CDM) - A description of the data, its
structure, allowable operations, and integrity constraints for
data of common interest within IISS.

Common Data Model Runtime (CDMR) - The collection of subroutines
and system processes that allow an AP to access the integrated
data base.

CS-ES Transformer - A CDMR precompiler generated software module
that performs the task of transforming data retrieved from the
Conceptual Schema format to the External Schema format.

Digital Equipment Corporation (DEC) - The minicomputer company
which manufactures, among others, the VAX series of computers.

Distributed Request Supervisor (DRS) - An IISS system subroutine that initiates and coordinates all the activity required to execute an NDML command. The DRS activates and sends parameters to RPs and Aggregators, and initiates file transfers.

International Business Machines Corporation (IBM) - The company that manufactures the testbed IBM 4381.

Integrated Data Base - The apparent relational data base that IISS can access. The integrated data base represents the program view of data including its structure, content, and allowable function.

Integrated Information Support System (IISS) - A product of the ICAM Integrated Data Base Management development program which is designed to connect heterogeneous computers and provide for distributed real-time data base access.

Multiple Virtual Storage/eXtended Architecture (MVS/XA) - An operating system that executes on large IBM computers.

Neutral Data Manipulation Language (NDML) - A nonprocedural data base manipulation language used to manipulate the data of the integrated data base. NDML statements (commands) are translated into DRS calls by an IISS utility called the precompiler (which builds the RPs at the same time).

Rehost - To port IISS software to a previously unsupported host computer, verify the transportability of IISS code on the new host, resolve incompatibilities, and develop system primitives as required.

Request Processor (RP) - An IISS system process which interfaces to one of the data bases IISS may access, and processes the parts of NDML commands which apply to that data base.

Virtual Address Extension (VAX) - A type of minicomputer manufactured by DEC which is used at the testbed.

Virtual Memory System (VMS) - An Operating System that executes on DEC VAX computers and is used on the testbed VAX.

SECTION 3

REQUIREMENTS

3.1 Computer Program Definition Requirements

The File Input/Output Primitives (FIOPs) are defined as a group of functions to allow the CDMR programs a generic interface to access system specific disk files. The functions include a Temporary File Namer, Open file, Read record, Write record, Seek forward and backward one record, Close file, and a limited Sort/Merge. These functions will be supported for sequential file types with fixed length records.

The CDMR programs exist only on the VAX and currently use system dependent file processing calls for their functions. The rehost effort to the IBM requires the FIOP interface to be done in another language compatible with COBOL calling conventions. They should also be as efficient as possible. For these reasons, the FIOPs will be written in the C programming language on the VAX system, and in Assembler language on the IBM.

3.2 General Interface Requirements

The CDMR and other IISS component systems will access sequential fixed-record-length disk files via the FIOPs. The only requirement between these programs is that the calling conventions of the language be "by reference" (address) and not "by value". The calling interface specifications are next in this document.

3.3 Detailed Interface Requirements

3.3.1 Parameter Formats

Character strings are arrays of characters with fixed maximum lengths, terminated by a NUL character ('LOW-VALUE' in COBOL, '\0' in C) if less than the maximum length. Address parameters are pointers to variables, and integer parameters are binary signed integers, a least 32 bits wide (PIC S9(9) COMP in COBOL, long int in C).

Note that in the C examples to follow, character arrays are one longer than needed, and a NUL is assigned to the last character for ease of use in C functions. This is not necessary, but many C string functions require a terminating NUL. As a reminder, arrays in C start with element zero [0] and the last subscript is one less the declared size.

3.3.2 Status Return Codes/Error Handling

Status return codes in the IISS are defined as strings (arrays) of 5 characters. A return code of all zeros ("00000") indicates successful completion of the called function, otherwise

an error has occurred. The error codes returned are formatted in
such a way that one can distinguish the subsystem, function being
performed, and have an idea about what kind of error occurred.
The FIOPs (like other primitives) will call ERRPRO with an
accompanying descriptive message to be put in the ERRLOG file if
the error is fatal. If the error code is a warning, ERRPRO will
not be called.

The left-most two characters of the returned error codes for
FIOPs are defined to be "11" (one-one). The left "1" indicates
"Primitives", and the right "1" shall indicate "File I/O"
primitives. The middle character will refer to the FIOP function
performed, and the right-most two digits will refer to function
specific errors. Further, the right-most digit will indicate the
severity of the error: zero will be for warnings, and nonzero will
be for fatal errors. This will allow the programmer to check this
right-most digit and easily determine a fatal/nonfatal status.

3.3.3 NAMFIL - Temporary File Name Function

Create and return a name for an IISS file, suitable for use
with the Open function. These file names will not match any
currently existing on the computer.

Usage Format:

COBOL Example:

```
01  FILE-NAME        PIC X(80).
    :
    :
    CALL "NAMFIL" USING FILE-NAME.
```

C Example:

```
char file_name[81];
    :
file_name[80] = '\0';
namfil(file_name);
```

Parameter:

File Name - A fully-qualified file name returned in local
system format, for a temporary file. The receiving variable
should be at least 80 bytes long. If the filename is less than 80
characters, it will be followed immediately by a NUL, and the rest
of the field will be undefined. If an error occurs, the first
position of this field will be returned as NUL.

3.3.4 OPNFIL - File Open Function

Prepare the file for access, initialize the file control
block (FCB), and allocate space for the file if needed. The file
status will be defined as "share" for input mode, and "exclusive"
for output or append modes. If a nonexistent file is opened for

append access, it will be created. In the unusual case that a
file is created for append, then a valid record length must be
passed to OPNFIL.

Usage Format:

COBOL Example:

```
01  FILE-NAME             PIC X(80)       VALUE SPACES.
01  FILE-CONTROL-BLOCK1   PIC S9(9) COMP VALUE ZERO.
01  RET-STATUS            PIC X(5)        VALUE SPACES.
 88  STATUS-OK                            VALUE "00000".
 88  NEW-APPEND-FILE                      VALUE "11130".
01  STATUS-RDF REDEFINES RET-STATUS.
 05  FILLER               PIC X(4).
 05  SEVERITY             PIC X.
     88  WARNING                  VALUE "0".
01  ACCESS-MODE           PIC X           VALUE "W".
01  RECORD-LENGTH         PIC S9(9) COMP VALUE ZERO.
01  NUMBER-OF-RECORDS     PIC S9(9) COMP VALUE ZERO.
:
:

   MOVE (expression) TO RECORD-LENGTH.
   MOVE (expression) TO NUMBER-OF-RECORDS.
   CALL "OPNFIL" USING FILE-CONTROL-BLOCK1, RET-STATUS,
                  FILE-NAME, ACCESS-MODE, RECORD-LENGTH,
                  NUMBER-OF-RECORDS.
   IF NOT WARNING
        PERFORM    ...(fatal status processing).
```

C Example:

```
#define GDSTAT    "00000"
#define WARNING   '0'
#define SEVERITY status[4]
int *fcb;
char status[6], file_name[81];
long rec_lth, num_recs;
:
/* Make sure string is NUL terminated */
status[5] = '\0';
:
rec_lth = (expression);
num_recs = (expression);
opnfil(&fcb, status, file_name, "W", &rec_lth, &num_recs);
if (SEVERITY != WARNING )
{       ...error processing;         )
```

Parameters:

File Control Block (FCB) - The address returned to the caller
when an FCB is created and initialized by Open. The FCB will have
to be passed to the other FIOPs by the caller. The format of the
control block is system dependent and the user need not be
concerned with its format. A separate FCB is required for each
file open at the same time.

Status - A generic IISS code returned from this function which is 5 characters long. Possible IISS codes are:

```
"00000" - SUCCESS
"11101" - INVALID PARAMETER
"11102" - FILE OPEN ERROR
"11103" - FILE NOT FOUND
"11104" - FILE NOT CREATED
"11105" - READ ACCESS DENIED
"11106" - WRITE ACCESS DENIED
"11107" - FILE NOT SEQUENTIAL
"11130" - CREATED NEW FILE FOR APPEND
```

File Name - A fully-qualified file name in local system format. A local file name is an array of characters, and if less than the maximum length is terminated by a space and/or NUL character. This filename may be user specified or one returned from NAMFIL.

Access Mode - Indicates whether the file is opened for reading (input only), writing (output only, all prior data lost), or append (output to the end of the file only, all prior data preserved). Legal values are "R", "W", and "A" for Read, Write, and Append.

Record Length - An integer number indicating the maximum size of the records in the file. The record buffer should be at least this size. This value is user specified for output and append access modes, and returned for input access. This field is checked for accuracy (or used as the required length to create a file) in append mode.

Number of Records - An integer that is the estimated total number of records for an output or append file given by the user. This parameter will allow OPNFIL to make sure that enough space exists for the file, and possibly to reserve it for use by this process. This parameter will not be used for Read access. The FIOPs will attempt to give the user more space for a file if required.

3.3.5 INPFIL - File Read Function

Retrieves data from an open file. Successive calls will read through the file in sequential order. This function will do record input only, so that if the size of the actual record is greater than the supplied buffer length, the record is truncated to this length. This should help prevent other data in the program from being inadvertently overwritten. To move the record pointer to the previous or next record without reading, use the seek function.

Usage Format:

    COBOL Example:

        CALL "INPFIL" USING FCB-X, RET-STATUS, RECORD-BUFFER,
                      BUFFER-LENGTH, RETURN-LENGTH.

    C Example:

    recbuf_lth = sizeof(rec_buf);
    inpfil(&fcb_x, status, rec_buf, &recbuf_lth, &rtn_lth);

Parameters:

    File Control Block (FCB) - A user supplied address, returned
previously by the Open routine.

    Status - A generic IISS code as mentioned previously.
Possible IISS codes are:

    "00000" - SUCCESS
    "11201" - INVALID PARAMETER
    "11202" - FILE NOT OPENED FOR INPUT
    "11203" - READ ERROR
    "11210" - END OF FILE
    "11240" - RECORD TRUNCATED

    Record Buffer - A buffer where the record data will be
stored. The record length was previously returned by the Open
call. The returned data is not NUL terminated.

    Buffer Length - The user specified integer length of the
buffer used for the returned record. If the user gives a length
smaller than the record length returned by the Open function, the
data will be truncated.

    Return Length - The actual integer size of the record placed
in the buffer by this function. It should always be equal to the
record length returned by the Open function for fixed record
length files, unless the buffer length is smaller than the record
size, in which case the return length will be set to the buffer
length. If the buffer length is smaller than the actual record
length, the data will be truncated and an appropriate warning
status code returned.

3.3.6 OUTFIL - File Write Function

    Add a new record to a file opened for output or append. The
record size must be the same as that referenced in the Open
function. File overflow will not be allowed, a "file full" error
will be returned instead.

<u>Usage Format</u>:

COBOL Example:

CALL "OUTFIL" USING FCB-X, RET-STATUS, RECORD-BUFFER,
            RECORD-LENGTH.

C Example:

outfil(&fcb_x, status, rec_buf, &rec_lth);

<u>Parameters</u>:

File Control Block (FCB) - A user supplied address, returned previously by the Open routine.

Status - A generic IISS code as mentioned previously. Possible IISS codes are:

"00000" - SUCCESS
"11301" - INVALID PARAMETER
"11302" - FILE NOT OPENED FOR OUTPUT OR APPEND
"11303" - FILE FULL
"11304" - WRITE ERROR
"11305" - RECORD TOO LARGE, NOT WRITTEN

Record Buffer - The record will be taken from this buffer and written to the file. The data in this record is not NUL terminated, and is assumed to be the length returned by the previous Open function.

Record Length - The integer length of the record. This parameter will be ignored for the current implementation of fixed length records. It will be the length determined in the Open function. Variable length output may be a future enhancement.

3.3.7 <u>SEKFIL - File Seek Function</u>

Repositions to a different record in the file relative to the current file position. This will allow very limited non-sequential access. Only two possibilities will be supported, seek to the next record and seek back one to the previous record. It is defined that a read moves the file position pointer to the beginning of the next record so that a subsequent forward seek will skip a record. <u>Seek is only valid for files opened for read.</u>

Usage Format:

COBOL Example:

```
MOVE 1 TO RECORD-COUNT.
CALL "SEKFIL" USING FCB-X, RET-STATUS, RECORD-COUNT.
```

C Example:

```
rec_cnt = 1;
sekfil(&fcb_x, status, &rec_cnt);
```

Parameters:

File Control Block (FCB) - A user supplied address, returned previously by the Open routine.

Status - A generic IISS code as mentioned previously. Possible IISS codes are:

```
"00000" - SUCCESS
"11401" - INVALID PARAMETER
"11402" - FILE NOT OPENED FOR INPUT
"11403" - FILE SEEK ERROR
"11410" - END OF FILE
"11420" - BEGINNING OF FILE
```

Record Count - This is a signed integer number of records to reposition in the file. Only two values will be accepted. These values are +1 for seeking forward one record and -1 to go back one record. Seeking forward (+1) several times will sequentially go forward one record at a time, as minus one (-1) will go back (backspace) one at a time. This means that after a record has just been read, you can seek -1 and issue a read to get that record data again.   Seek -1 twice and read would be required to get the record previous to the one just read.

3.3.8   CLSFIL - File Close Function

Terminate file access, cleanup (release resources, etc.), keep or delete the file as user specified.

Usage Format:

COBOL Example:

```
CALL "CLSFIL" USING FCB, RET-STATUS, DISPOSITION.
```

C Example:

```
clsfil(&fcb_1, status, "K");
```

Parameters:

File Control Block (FCB) - A user supplied address, returned previously by the Open routine.

Status - A generic IISS code as mentioned previously. Possible IISS codes are:

"00000" - SUCCESS
"11701" - INVALID PARAMETER
"11702" - FILE NOT DELETED
"11703" - FILE NOT CLOSED

Disposition - A one character string which indicates file disposition. There are three options: "K", "P", and "D" for Keep, Pass, and Delete. A file closed with Keep disposition will be saved to disk. A file closed with Pass will be held (possibly in memory) for future access until another process deletes it or makes it permanent. Delete causes the file to be deleted (erased). If this parameter is not specified (left blank or NUL) then Pass will be assumed.

3.3.9 SRTFIL - Sort/Merge Function

Take one or more files, merging if more than one, and order the records according to the users' directions. The result will be put in an output file (name supplied by the user) different from any of the input file(s). The user is responsible for making the fields in the input files compatible. The sort routine may pad records so that all output records are the same length. If the output file already exists, it will be overwritten.

Usage Format:

COBOL Example:

```
CALL "SRTFIL" USING IN-FILE-NAMES, OUT-FILE-NAME, SORT-KEY,
              OPTIONS, OUT-RECORD-COUNT, RET-STATUS.
```

C Example:

```
srtfil(in_fnames, out_fname, sort_key, options, &out_cnt,
    status);
```

Parameters:

Input File Names - A string containing one to four file names (multiple file names are separated by a plus sign) indicating the file(s) to be sorted. The total length of this field will be a maximum of 324 characters. There must be a terminating NUL characterafter the last input filename given. Extra spaces around separate file names will be ignored.

Output File Name - A string containing one file name in the same format shown for the Temporary File Name and Open functions. If this file already exists, it will be overwritten.

Sort Key Descriptor - A string describing the sort key. For each field chosen to sort upon, there will be four sub-parameters in the sort key separated by commas. The sort control groups (sets of 4 sub-parameters), if more than one, must be specified from highest priority to lowest and are also separated by commas. There shall be no limit on the number of sort control groups, except that the maximum length of the sort key descriptor string will be 2000 characters. The end of the specifications will be indicated by either a zero in sub-parameter 1, a blank, or a NUL character. The sub-parameters are: (1) starting position of the field, (2) length of the field in bytes, (3) type of field, and (4) the sort order.

The starting position and field length are positive integer numbers. The field type is a two character field and has valid values of:

CH - character, unsigned
BI - binary, unsigned
PD - packed decimal, signed
DT - signed decimal, trailing overpunch

The sort order is either A for Ascending, or D for Descending.

An example of a sort key descriptor might be:

"1,8,CH,A,9,4,PD,D"

which indicates that the first sort field is composed of the first eight characters of the record in ascending order and then a 4 byte packed decimal field starting at the ninth character in descending order.

NOTE: If the first character of the sort key descriptor is NUL, blank or zero, a file copy (concatenate if more than one input file) will be assumed.

Sort Options - The sort options string will be reserved for future use if required. Specific defaults will be assumed in this implementation. They are: (1) the character collating sequence will be the computer's native code, ASCII on the VAX, and EBCDIC on the IBM, (2) records with duplicate keys will be kept in the resultant file, (3) the order of equally collating records is not guaranteed to be preserved from input to output.

Output Record Count - This is the integer number of records that the output file contains upon successful sort completion.

Status - A generic IISS code as mentioned previously.
Possible IISS codes are:

```
"00000" - SUCCESS
"11901" - INVALID PARAMETER
"11902" - SORT PACKAGE ERROR
"11903" - ERROR CLOSING INPUT FILE(S)
"11904" - ERROR CLOSING OUTPUT FILE
"11905" - ERROR OPENING INPUT FILE(S)
"11906" - ERROR OPENING OUTPUT FILE
"11907" - ERROR READING FILE(S)
"11908" - INSUFFICIENT WORK SPACE
"11909" - ERROR WRITING FILE
```

3.4 Executable Code Creation

Executable code will be created on the IISS testbed VAX using the VAX-11 C compiler. IBM system specific code will be created on the IISS testbed IBM using its native Assembler language. After unit testing, the FIOP functions will be used and tested by the CDMR programs. After the CDMR rehost is complete, the FIOPs will be released for integration testing with the CDMR.

3.5 CPCI Design Description

This section contains the detailed technical descriptions of the File I/O Primitives for sequential fixed-record-length files.

There are seven File I/O Primitive routines: "NAMFIL", "OPNFIL", "INPFIL", "OUTFIL", "SEKFIL", "CLSFIL", and "SRTFIL". The FIOPs shall use the preexisting system dependent routines that exist on the VAX and the IBM computers for accessing files and sorting the data in them. The FIOPs basically conform to the normal file operations described in any programming language, but have two additions. The Temporary File Name function and the File Sort/Merge are actually file utilities. A high level English algorithm for each of the functions follows.

NAMFIL - Temporary File Name Function:

1) Set the filename to NUL.
2) Lock on a resource so that this function will complete before being used again.
3) If unsuccessful lock, return.
4) Generate a file name from fixed and variable parts to guarantee uniqueness on the resident computer. The filename must be identified somehow with the IISS.
5) Unlock the resource. If unsuccessful unlock, return.
6) Copy the filename to the given parameter and return.

OPNFIL - Open a File:

1) Check parameters for validity. If any invalid, do error processing and return error.
2) Allocate and clear required control block(s).
3) Set up file control block(s) with required data.
4) Open the file with the access mode and allocation size specified. Set up so that a nonexistent file opened for append will be created.
5) If open fails, free used memory space, do error processing and return error.
6) Check for correct file type and record size. If incorrect, do error processing and return appropriate error.
7) Add file control block(s) to internal open list.
8) Return file control block pointer, zero status, and record length if the access mode is input or append.

INPFIL - Read a Record From a File:

1) Check parameters for validity. If any invalid, do error processing and return error.
2) Check access mode. If not opened for Read, do error processing and return error.
3) Read next record at the current file position. If End of File, return EOF status.
4) If read error, do error processing and return error.
5) Update file position to the next record where required.
6) Copy buffer-length amount of data to the buffer given by the caller.
7) Return record length if it is less than or equal to the buffer length, and the buffer length if greater. Return status of "Record Truncated" if record length is greater than buffer length, zero status otherwise.

OUTFIL - Write a Record to a File:

1) Check parameters for validity. If any invalid, do error processing and return error.
2) Check access mode. If not opened for Write or Append, do error processing and return error.
3) Write the record to the file. If write error or file is full, do error processing and return error.
4) Return successful status.

SEKFIL - Seek Forward or Backward One Record:

1) Check parameters for validity. If any invalid, do error processing and return error.
2) Make sure file was opened for input. If not, do error processing and return error.
3) If not at beginning or end of file, update file position.
4) If file position is now at beginning or end of file, return BOF or EOF status code.
5) Return successful status.

CLSFIL - Close a File:

1) Check parameters for validity. If any invalid, do error processing and return error.
2) Close the file with disposition requested. If file not closed, do error processing and return error.
3) If disposition is Delete, delete the file. If file couldn't be deleted, do error processing and return error.
4) Remove file control block(s) from open list and free resources.
5) Return successful status.

SRTFIL - Sort/Merge File(s):

1) Check parameters for validity. If any invalid, do error processing and return error.
2) Build (translate) parameters for sort package.
3) Invoke sort package with user specified and other default parameters.
4) If sort package or other file error, do error processing and return error (output file is deleted).
5) Return number of records in output file and successful status.

3.5.1 Data Organization

The following is a summary description of the parameters used by the FIOP functions. Other data elements within each of the VAX and IBM specific routines are not included here.

Parameter	Type	Length/Range	Used by
ACCESS-MODE	string	1 character	OPNFIL
BUFFER-LENGTH	integer	1 to rec.lth.	INPFIL
DISPOSITION	string	1 character	CLSFIL
FCB	pointer		OPNFIL, INPFIL, OUTFIL, SEKFIL, CLSFIL
FILENAME	string	80 characters	NAMFIL, OPNFIL
IN-FILE-NAMES	string	324 characters	SRTFIL
NUMBER-OF-RECORDS	integer	> 0	OPNFIL
OPTIONS	string	not used	SRTFIL
OUT-FILE-NAME	string	80 characters	SRTFIL
OUT-RECORD-COUNT	integer	>=0	SRTFIL
RECORD-BUFFER	string	user specified	INPFIL, OUTFIL
RECORD-COUNT	integer	+1, -1	SEKFIL
RECORD-LENGTH	integer	> 0	OPNFIL, OUTFIL
RETURN-LENGTH	integer	> 0	INPFIL
SORT-KEY	string	2000 characters	SRTFIL
STATUS	string	5 characters	OPNFIL, INPFIL, OUTFIL, SEKFIL, CLSFIL, SRTFIL

3.5.2 Limitations

The File Input/Output Primitives will be limited for use with sequential disk files with fixed length records. They cannot be used with any tape files. The hosts upon which the FIOPs will exist are the VAX and IBM IISS testbed computers using the VMS and MVS/XA operating systems respectively. Initial usage of the FIOPs will be by the CDMR, so the design constraints were dictated by the requirements of the CDMR with consideration given to future expansion.

3.5.3 Listings

To be supplied later in an appendix.

SECTION 4

QUALITY ASSURANCE PROVISIONS

4.1 Introduction and Definitions

"Testing" is a systematic process that may be planned and explicitly stated. Test techniques and procedures may be defined in advance and a sequence of test steps may be specified.   "Debugging" is the process of isolation and correction of the cause of an error.

"Antibugging" is defined as the philosophy of writing programs in such a way as to make bugs less likely to occur and, when they do occur, to make them more noticeable to the programmer and the user. In other words, each routine should perform as much error checking as is practical and possible.

4.2 Computer Programming Test and Evaluation

The quality assurance provisions for testing will consist of the normal testing techniques that are accomplished during the construction process. They consist of design and code walk-throughs, unit testing, and integration testing. These tests will be performed by the design team. Final integration testing will be performed by the CDMR rehost team. Structured design, design walk-throughs, and the incorporation of "antibugging" facilitate this testing by exposing and addressing problem areas before they become coded "bugs".

Unit testing will be performed on both of the IISS testbed computers with the same test programs. COBOL test programs will be used to simulate the calls by the higher level CDMR to the low-level FIOP routines.

Integration testing will entail actual use by the rehosted CDMR on both computers of the IISS testbed.   Prior to integration testing on the IBM, the FIOP routines will be incorporated in the IISS environment similar to the currently existing Inter-Host Communication (IHC) Primitives and Inter-Process Communication (IPC) Primitives.

SECTION 5

PREPARATION FOR DELIVERY

The initial implementation site for the constructed software will be the IISS VAX and IBM testbed computers located at Arizona State University in Tempe, Arizona. Preparation for delivery will be incorporated by the standard IISS release procedure contained in the "Software Configuration Management (SCM) Administrator's Manual" [8].

U.S. Government Printing Office 648-127